FUNCIONALIDADES DE SOPORTE EN RDBMS:

BLOQUEOS (LOCKS)

Llevemos la teoría a la práctica con MySQL

Introducción:

Los mecanismos de bloqueo son críticos para mantener la base de datos **consistente**. Sin embargo, los bloqueos innecesarios pueden tener un impacto **negativo** en el rendimiento del sistema. Por lo tanto, es crucial para cualquier administrador comprender los principios del bloqueo, así como saber **cuándo, dónde y cómo** adquirir un bloqueo.

Si bien en el manual de referencia de *MySQL* hay varios tipos de bloqueos, vamos a profundizar en dos de ellos (que además son los esenciales):

- Bloqueo de lectura/compartido (**Shared lock**): Estos bloqueos son necesarios para proporcionar lecturas consistentes al no permitir operaciones de escritura en este objeto. Mientras una transacción mantiene el bloqueo de lectura, no se puede adquirir el bloqueo de escritura en ese objeto, lo que coloca la transacción en la cola de espera.
- Bloqueo de escritura (**Exclusive lock**): Estos bloqueos son necesarios cuando un objeto necesita ser modificado. Solo puede haber un bloqueo de escritura de un mismo objeto al mismo tiempo. Mientras una transacción mantiene el bloqueo de escritura en un objeto, todas las demás solicitudes de bloqueo deben esperar.

Ambos bloqueos son necesarios para que una base de datos pueda ser **consistente**, ya que imaginemos el caso de que un usuario quiera acceder a la información de un cliente y otro usuario a su vez, quiera modificar la información de ese cliente. En estos casos, los bloqueos son necesarios para que todos puedan acceder a los datos verdaderos y actualizados.

También cabe destacar, que el bloqueo de escritura tiene una prioridad más alta que el bloqueo de lectura. Cuando se desbloquea un objeto, y si hay solicitudes de bloqueo de escritura esperando en la cola se le da prioridad antes de los de lectura.

Bloqueos en InnoDB:

InnoDB admite bloqueo a nivel de tabla y a nivel de fila.

El bloqueo a nivel de fila ofrece un buen rendimiento y es muy útil para grandes cargas de trabajo, en las que varias transacciones se leen y/o escriben en la misma **tabla** simultáneamente, pero escriben en filas distintas, sin obligarse mutuamente a esperar.

Para comprender los conceptos de bloqueo, crearemos la base de datos “db_name”, la tabla “db_table” y agregaremos una fila de la siguiente forma:

```
1  CREATE DATABASE db_name;
2
3  CREATE TABLE db_name.db_table (
4    id int NOT NULL AUTO_INCREMENT,
5    campo1 varchar(128) NOT NULL,
6    PRIMARY KEY (id)
7  )ENGINE=innodb;
8
9  INSERT INTO db_name.db_table (campo1)
10 VALUES('fila1');
11
12
```

(Ya que, por default, *MySQL* si no especificamos qué motor usar nos selecciona InnoDB, pero es una buena práctica especificar qué motor queremos usar, simplemente para estar seguros).

Bloqueos de Filas:

Existen 2 formas de poder ejecutar un bloqueo explícito de filas en InnoDB:

- FOR UPDATE
- LOCK IN SHARED MODE

Cualquier bloqueo retenido con la cláusula **FOR UPDATE** no permitirá que otras transacciones lean (utilizando la cláusula FOR UPDATE), actualicen o eliminen las filas hasta que la transacción se confirme (commit) o revierta (rollback), liberando el bloqueo. Esto es básicamente un bloqueo de escritura (**Exclusive lock**).

Para poder ver esto, vamos a necesitar tener dos sesiones (instancias) abiertas conectadas a la misma base de datos, entonces en la primera sesión hacemos una lectura con el bloqueo *FOR UPDATE* de la siguiente forma:

```
1  START TRANSACTION;
2  SELECT id, campo1
3  FROM db_name.db_table
4  WHERE id=1
5  FOR UPDATE;
6
7
```

Result Grid | Filter Rows:

id	campo1
1	fila1
NULL	NULL

Como podemos ver, estamos haciendo una lectura con un bloqueo de escritura, esto significa que ninguna otra transacción puede leer/modificar esa fila hasta que el bloqueo termine (ya sea por un **COMMIT** o un **ROLLBACK**). Pero ¿qué sucede si queremos forzar esa lectura? Para eso usamos otra sesión (no importa el usuario, pueden ser ambos usuarios root, sin problemas) y probamos la misma transacción que hicimos anteriormente:

✓	1	10:20:02	START TRANSACTION	0 row(s) affected
4	2	10:20:02	SELECT id, campo1 FROM db_name.db_table WHERE id=1 FOR UPD...	Running...

Como era de esperar, quedaremos en espera de que se termine la transacción que está ocupando esa fila para poder hacer la lectura/escritura que queremos. Mientras esté el bloqueo *FOR UPDATE*, tampoco vamos a poder borrar esa fila.

En el caso de *LOCK IN SHARED MODE* cualquier bloqueo retenido con esta cláusula, permitirá que otras transacciones lean la fila bloqueada, pero no permitirá que otras transacciones escriban en la fila hasta que la transacción se confirme o se revierta y se libere el bloqueo. Esto es básicamente un bloqueo de lectura/compartido (**Shared lock**).

```
1  START TRANSACTION;
2  SELECT id, campo1
3  FROM db_name.db_table
4  WHERE id=1
5  LOCK IN SHARE MODE;
6
7  UPDATE db_name.db_table
8  SET campo1='fila3'
9  WHERE id=1;
```

✓	2	10:38:48	SELECT id, campo1 FROM db_name.db_table WHERE id=1 LOCK I...	1 row(s) returned
4	3	10:38:51	UPDATE db_name.db_table SET campo1='fila3' WHERE id=1	Running...

Como podemos ver, en la lectura compartida (ya en la segunda sesión) funciona correctamente, pero si queremos hacer luego un update de esa fila (que está con bloqueocompartido por la primera sesión) volvemos a estar en cola de espera que el bloqueo se libere mediante un **COMMIT** o un **ROLLBACK**.

Bloqueos de Tablas:

En InnoDB, las transacciones también pueden adquirir bloqueos a nivel de tabla. Para adquirir un bloqueo de tabla, podemos usar la siguiente instrucción:

LOCK TABLES <*nombre-de-la-tabla*> [**READ | WRITE**]

Para liberar un bloqueo de tabla, debemos ejecutar la siguiente instrucción:

UNLOCK TABLES

El modo de bloqueo READ / WRITE es similar a los explicados anteriormente. Probemos cada modo de bloqueo:

Bloqueo de tabla para escritura

Tenga en cuenta que solo la transacción que contiene el bloqueo de la tabla WRITE puede leer y escribir datos de la tabla. Las otras transacciones no pueden leer y escribir desde la tabla hasta que se libere el bloqueo de la tabla.

```
1  START TRANSACTION;
2  LOCK TABLE db_name.db_table WRITE;
3
4  SELECT  id, campo1
5  FROM db_name.db_table;
6
7  INSERT INTO db_name.db_table (campo1)
8  VALUES('fila2');
9
10
```

✓	1	10:53:53	START TRANSACTION	0 row(s) affected
✓	2	10:53:53	LOCK TABLE db_name.db_table WRITE	0 row(s) affected
✓	3	10:53:53	SELECT id, campo1 FROM db_name.db_table	1 row(s) returned
✓	4	10:53:54	INSERT INTO db_name.db_table (campo1) VALUES('fila2')	1 row(s) affected

Mientras hagamos en la misma transacción del bloqueo, vamos a poder ejecutar todas las acciones, ahora hagamos lo mismo en una segunda sesión:

```
START TRANSACTION;
SELECT  id, campo1
FROM db_name.db_table;

START TRANSACTION;
INSERT INTO db_name.db_table (campo1)
VALUES('fila3');
```

✓	1	11:00:33	START TRANSACTION	0 row(s) affected
4	2	11:00:33	INSERT INTO db_name.db_table (campo1) VALUES('fila3')	Running...

En ambos casos, quedamos a la espera que el bloqueo se libere para poder proseguir con la transacción.

Bloqueo de tabla para lectura

Abrimos la primera sesión y ejecutamos lo siguiente:

```
START TRANSACTION;
LOCK TABLE db_name.db_table READ;
```

El siguiente paso es en nuestra segunda sesión, ejecutar otra transacción tratando de insertar una nueva fila en la tabla y también leyendo esa tabla.

```
START TRANSACTION;

SELECT  id, campo1
FROM db_name.db_table;

INSERT INTO db_name.db_table (campo1)
VALUES('row1');
```

✓	1	11:07:11	START TRANSACTION	0 row(s) affected
✓	2	11:07:11	SELECT id, campo1 FROM db_name.db_table	2 row(s) returned
4	3	11:07:11	INSERT INTO db_name.db_table (campo1) VALUES('row1')	Running...

Podemos ver que la transacción está bloqueada al ejecutar la operación INSERT porque la primera transacción ya contiene el bloqueo de lectura, lo que no permite la escritura de ningún dato, pero si la lectura.

Para ambos casos, no olvidar de liberar las tablas, ya que sino no podremos utilizarlas normalmente.

```
UNLOCK TABLES;
```

Referencias:

- Capítulo 17.7.1 “Bloqueo InnoDB” del Manual de Referencia de MySQL 8.4: https://dev.mysql.com/doc/refman/8.4/en/innodb-locking.html
- Presentaciones de teoría de la materia (“FUNCIONALIDAD DE SOPORTE EN RDBMS”).